

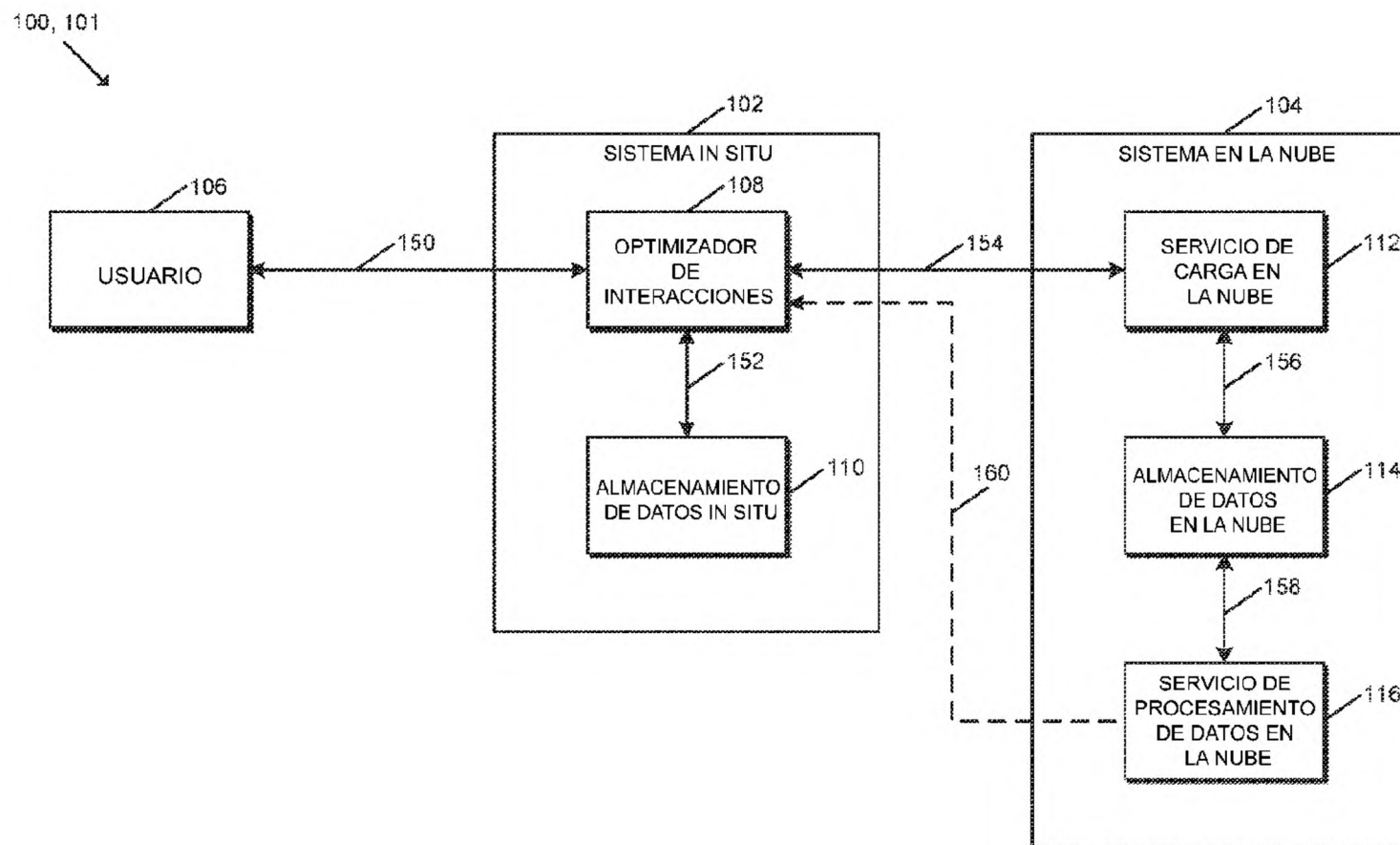


FIG. 1

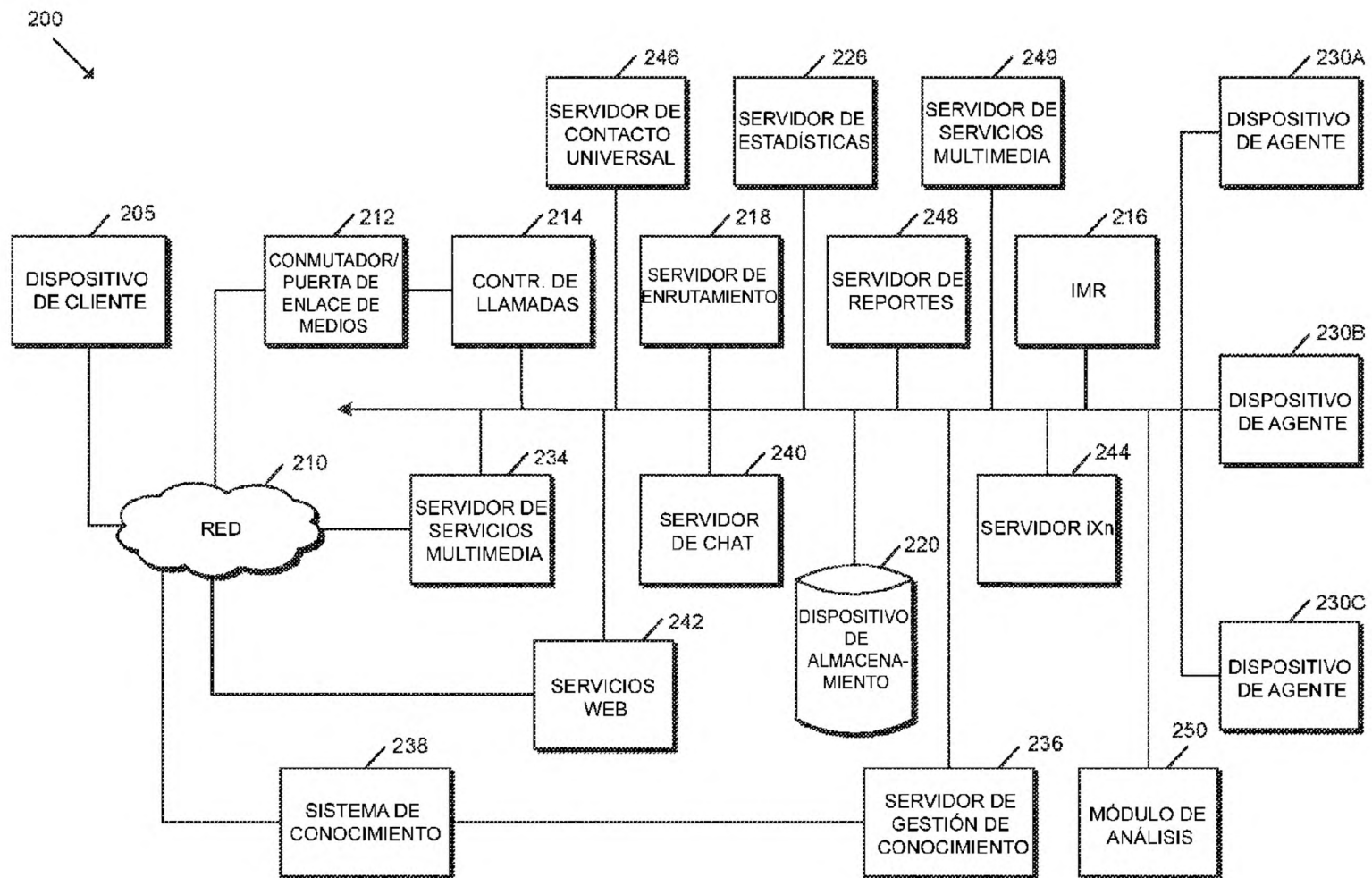


FIG. 2

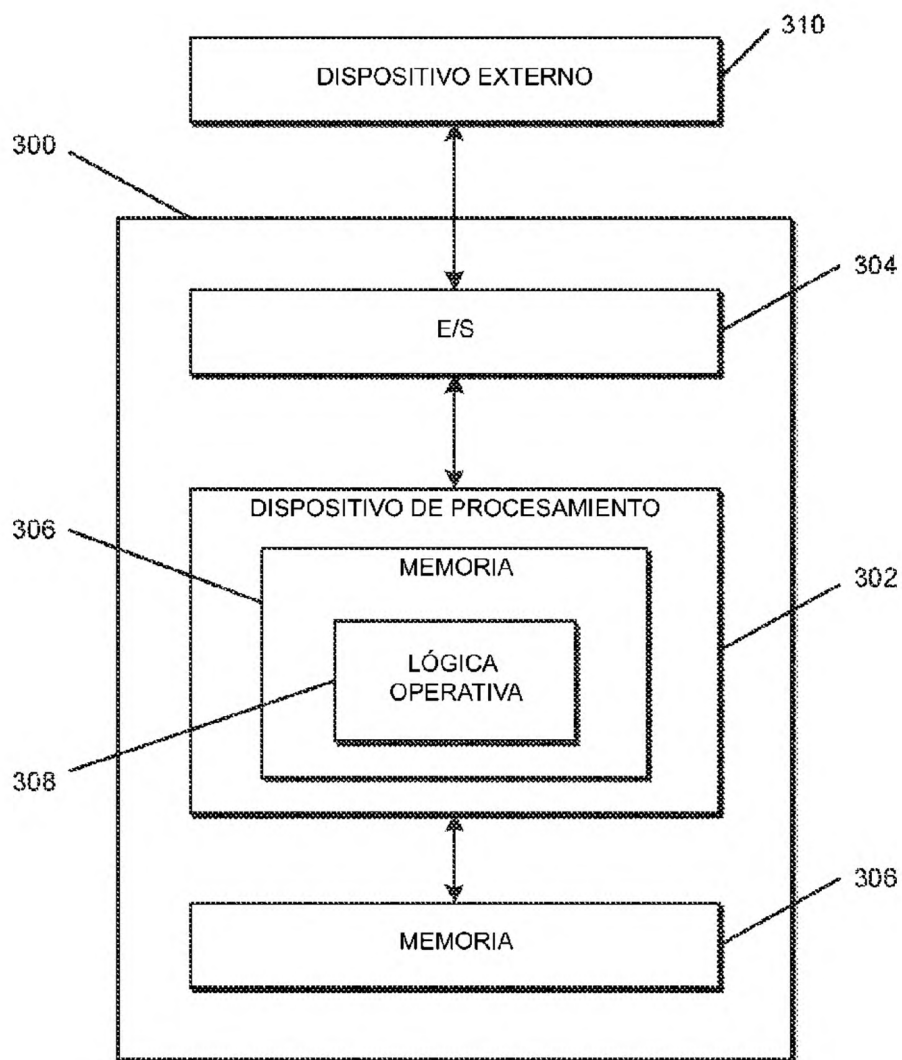


FIG. 3

```

lastInterval : 0
lastTime : statistics.firstItem.Time
for (cada ítem en la estadística)
{
    currentTime : item. Time
    currentInterval : Diferencia de tiempo de currentTime y last.Time
    if (currentInterval = 0)
    {
        //1.º valor - Buscar el ítem siguiente y obtener
        el intervalo siguiente
        //Usar el intervalo siguiente para calcular el número
        de interacciones por min y tratar de calcular el número
        de interacciones para el intervalo de 15 min.
        //Por ejemplo: Primer tiempo : 5:30 - Valor : 91 Tiempo
        siguiente : 6:00
        //Intervalo siguiente = 30 min. Por lo tanto el número de
        interacciones por
        min basado en el intervalo - 3,03
        //Valor = Núm. de interacciones por min * Timedifference en
        minutos o Núm. de interacciones por segundo * Timedifference
        en segundos //Valor = (Value / intervalInMin) *
        Timedifference en minutos o Núm. de interacciones por segundo
        * Timedifference en segundos
        //Valor = (91 / 30) * 15
        //Para 5:15 a 5:30 - Valor - 45,5
        nextTime : statistics.nextItem.Time
        nextInterval : TimeDifference de nextTime y currentTime
        valueToBe : (value/nextInterval) * 15 //Value = Núm. de
        interacciones por min basado en el intervalo siguiente * 15
        minutos (puesto que estamos tratando de rellenar los datos
        para cada intervalo de 15 minutos)
        result [floorCurrentTime] = valueToBe
    }
}

```

FIG. 4

```

else {
  if(lastInterval not equal to currentInterval.)
  {
    while (ceilLastTime < floorCurrentTime)
    {
      //Rellenar los espacios con -1. ¡Asegúrese de que no
      haya una entrada para el tiempo en el mapa de resultados!
      result [ceilLastTime] = -1
      ceilLastTime = NextTimeSeriesEntry of
      ceilLastTime
    }
    // Actualizar lastTime y currentInterval basados en
    el intervalo siguiente si hemos rellenado los espacios
    usando el bucle while anterior
    if the while loop has run
    {
      nextTime : Statistics.nextItem.Time
      nextInterval : TimeDifference of nextTime and
      currentTime
      lastTime: adjustTime (currentTime, - nextInterval)
      currentInterval : TimeDifference of currentTime
      and lastTime
    }
  }
  // el bucle do while se encarga de rellenar datos hasta
  el tiempo de la penúltima ronda
  // si lastTime = 6:20 y currentTime = 7:10
  // do while 100p rellena datos para 6:30, 6:45, 7:00
  lastTime1 : lastTime
  ceilLastTime1 : Ceil (lastTime1)

```

FIG. 5

```

do
{
    timeDiff : TimeDifference of ceilLastTime1 and
    lastTime1
    valueToBe : (value/currentInterval)*
    timeDiff //Valor = núm. de interacciones por
    segundo * Timedifference en segundos
    result[ceilLastTime1] = valueToBe //Si el valor para
    ceilLastTime1 ya se encuentra en el
    resultado, entonces sumar valueToBe al valor
    contra ceilLastTime en el resultado
    lastTime1 : ceilLastTime1
    ceilLastTime1: NextTimeSeriesEntry of lastTime1
}while (ceilLastTime1 < ceilCurrentTime)

// Más abajo la lógica rellena datos para 7:15
timeDiff : TimeDifference of currentTime and lastTime1
valueToBe : (value/currentInterval)* timeDiff //Valor =
núm. de interacciones por segundo * Timedifference
en segundos
result (ceilLastTime1] = valueToBe //Si el valor para
ceilLastTime1 ya se encuentra en el resultado, entonces
sumar el valor valueToBe al valor contra ceilLastTime en
el resultado
}
lastTime : currentTime
lastInterval : currentInterval
}

```

FIG. 6

```

//Los datos involucrados en el pronóstico tendrán un límite
superior para interpolación, que se indicará como ILimit.
Estos podrían ser minutos, horas, días, etc.
//Los datos involucrados en el pronóstico tendrán una longitud
de intervalo, que se indicará como ILength. Estos podrían ser
minutos, horas, días, etc. Sin embargo, tienen las
restricciones de ILength < ILimit.
//Los datos específicos para nuestro caso de uso utilizan
una ILength = 15min y un ILimit = 1hora.
//Para nuestros datos, todos los buckets se identifican en
intervalos de 15 minutos que comienzan en 00 minutos. Esto
implica que los buckets son 00, 15, 30, y 45.

Algoritmo:
//Entrada: (1)Rango de datos (2) Datos reales para el rango
deseado
: Identificar el tiempo de comienzo de los datos
:Rellenar desde el tiempo de inicio deseado al tiempo de
inicio de datos con valores vacíos

//Todos los buckets miran hacia atrás en el tiempo, por lo
tanto el bucket ILength1 es (ILength0, ILength1]
//Para nuestros datos bucket 15, son datos desde (00, 15], donde
el índice de tiempo previo es no inclusivo.
:Alinear el primer punto de datos al primer bucket ILength.
//Para nuestros datos esta alineación es en el primer bucket
de 15 minutos

: For (todos los datos reales)
//Se necesita Backfilling cuando los datos no se reparten
uniformemente en las barreras de ILength
//Para nuestros datos se utilizan barreras de 15 minutos. Por
ejemplo, los datos que rompen una barrera extendiéndose desde
05 a 23 rellenarían los buckets (00,15] y (15,30]
::Rellenar desde el último intervalo rellenado
hasta (intervalo de datos actuales - ILimit) con ceros
::Desglosar datos sobre intervalos ILength comenzando con el
último intervalo no rellenado o parcialmente rellenado e
intervalo(s) subsiguiente(s) hasta agotar los datos.

```

FIG. 7